Iterative Recovery of Controllability via Maximum Matching*

Shuo Zhang[1] Stephen D. Wolthusen [2]

***Abstract*— Controllability is significant for dynamical systems, and iterative recovery of controllability is indispensable sometimes. We consider any large, sparse *Erdős-Rényi* random digraph with a linear time-invariant control model as an input graph, obtained by adding one node to an original random digraph, and we seek to recover its controllability via efficiently identifying a maximum matching of it rather than recomputation. Particularly, we assume that any input graph contains a known matching that is a maximum matching of the original random digraph. In our solution, we depend on a bipartite graph one-to-one mapped by an input digraph to find its augmenting paths relative to a matching corresponding to the known matching of the input graph. By finding augmenting paths within this mapped bipartite graph, we eventually find a maximum matching and recover the controllability of this input digraph in linear time as a result.**

I. INTRODUCTION

Complex networks at the influential position have been over decades [1] [2] [3] [4], which influence us to further control [5] and observe [6] networked systems. Because the linear time-invariant(LTI) systems could change over time due to malicious attack, random failure or insertion and deletion of new system nodes or edges, their digraphs would also be changed by additions or deletions of nodes or edges. Clearly, efficient recovery of controllability of a digraph in LTI systems after each change is essential, so that iterative recovery of controllability can be effectively executed. We solve the problem of recovery of controllability of a digraph after an addition of one node with the worse-case complexity. Our contribution is faster recovery of controllability in linear time compared with related work of [24] [25] [26] [27].

Based on the Kalman's control theory [7] and Lin's structural controllability [8], Liu $et\ al$ [5] raised the minimum input theorem to give a powerful mechanism of effectively obtaining the minimum number of external inputs to fully control a digraph in linear time-invariant systems via the maximum matching. A matching of a digraph is a set of arcs without sharing any common head or tail, when this set is not a subset of any other matching set, it is a maximum matching [9]. If a node is a head of an arc of a maximum matching, it is called a matched node based on this maximum matching, otherwise, it is unmatched.

According to the minimum input theorem [5], continuously or iteratively identifying a maximum matching of a digraph under malicious attack can be used to reflect the fluctucation of controllability during the period of deletions of nodes or edges and further quantitatively illustrate the robustness of network controllability against malicious attacks [10] [11] [12]. On the other hand, optimising network controllability [13] [14] also needs to iteratively calculating a maximum matching of the latest improved digraph to check whether the controllability is satisfying or not. However, any best-known maximum matching algorithms [15] [16] is not efficient to reuse over times. Thus, our problem proposed to be solved by efficiently finding a maximum matching of an digraph after addition of one node withour recomputation.

We assume that any given large, sparse *Erdős-Rényi* random input digraph of linear time-invariant dynamics is generated by adding one node and relative arcs into an original random digraph, where existence of edges between added node and the original digraph depends on the edge existence probability [17] of this original random digraph, and this original digraph also contains a known maximum matching. To find a maximum matching of an input digrph, we use a bipartite representation of it to find augmenting paths [18] [19] related to a matching mapped by the known one of the input graph. In the worst case, we could finally obtain a maximum matching of an input digraph, and recover its controllability in linear time, so as to iteratively recover controllability of each digraph after an addition of one node could be effectively operated.

Remaining paper is structured as follows: Sec.II shows related work about controllability and maximum matching problems. Sec.III illustrates the network controllability. Sec.IV introduces input graphs and augmenting paths. Sec.V recoveries controllability of an input digraph. Last section concludes this paper.

II. RELATED WORK

Controllability of LTI systems could be approached and recovered by several ways. One is by the maximum matching of the digraph of a LTI system [5] as a way to obtain complete controllability. And another way is by the power dominating set [20] to firstly obtain structural controllability and then acquire controllability in general cases. It is said that direct computation of power dominating set problem is not desirable in general graphs but $\Theta(log\ n)$-approximable [21], where n is the number of vertices. By the original research about power dominiting set [22], [23], Alwasel $et\ al.$ [24] [25] [26] recovered the approximated structural controllability [8] of the *Erdős-Rényi* random digraph after removing vertices by using power dominating set, while Alcaraz $et\ al.$ [27] also used a same approach to recover

[1] Shuo Zhang is with School of Mathematics and Information Security, Royal Holloway, University of London, Egham TW20 0EX, UK MYVA375@live.rhul.ac.uk
[2] Stephen D. Wolthusen with School of Mathematics and Information Security, Royal Holloway, University of London, Egham TW20 0EX, UK stephen.wolthusen@rhul.ac.uk

exact structural controllability of scale-free networks after nodal removal. By contrast, we are going to recover the network controllability by finding a maximum matching of a digraph in LTI system after adding one node.

Finding a maximum cardinality or weighted matching is related to the combinatorial optimization, which has a long history [28]. Given the static graphs, the best-known Hopcroft-Karp algorithm [15] of effectively identifying a maximum matching in the bipartite graph runs in $O(\sqrt{n}m)$, where n is the number of vertices of a graph, and m is that of edges. When the given bipartite is dense such as $m = n^2$, time complexity becomes $O(m\sqrt{n/logn})$ [29]. By contrast, Micali and Vazirani [16] found the maximum cardinality matching on the general graphs, and the worst-case execution time of their algorithm is also $O(\sqrt{n}m)$. Up to date, these two results are the bound of efficiently finding maximum matching by deterministic algorithms unless it allows approximation, which could be executed much faster than identifying an exact maximum matching on static graphs [30]. Since graphs would be changed along with the change of systems, it means that recovery of controllability might be iterative after each change. In this way, simply using these two algorithms or finding power dominating set for iteratively acquiring certain controllability is not efficient or even unrealistic for large digraphs.

Thus, we propose to efficiently identify a maximum matching of each digraph after adding a node without recomputation. This problem can be further classified into the dynamic graph problem [31], which mainly seeks to efficiently update a solution via maintaining a data structure after a change rather than recompute a solution from a change. A fully dynamic graph problem addresses the update operations of unlimited insertions and deletions of edges or vertices, while a partially dynamic only considers either insertions or deletions of edges or vertices. Fully dynamic approximate maximum-cardinality matching problem is popular in recent years. In 2010, Rubinfeld $et\ al.$ [32] designed a randomized algorithm that maintains a $O(1)$-approximation maximum matching in $O(log^2 n)$ time. Baswana, Gupta and Sen [33] then gave a 2-approximation maximum matching in a dynamic graph with $O(log\ n)$ amortized time. In [34], with a deterministic data structure, the approximation ratio is $(3/2 + \epsilon)$ and the worst case time complexity is $O(m^{1/4}\epsilon^{-2.5})(\epsilon > 0)$. Until now, [35] presented a deterministic data structure with $(2 + \epsilon)$-approximation and the worst-case time complexity is $O(log^3 n)$. Nevertheless, to derive an exact size of a maximum matching in the fully dynamic, the best known update time is $O(n^{1.495})$ [36]. In comparison, our problem is partially dynamic and each update only allows addition of one node. Besides, our algorithms are deterministic and we only concerns the worst-case complexity. We can efficiently derive a maximum matching of an input digraph in the linear time rather than recompute a new maximum matching by using an algorithm of either [15] or [16]. Once a maximum matching of an incremental digraph is found, its controllability could be also recovered according to the minimum input theorem [5].

III. Network Controllability

A dynamic system is controllable if this system can be driven from any initial state to any proposed final state by properly using external inputs within limited time [7] [18] [37]. In general, linear time-invariant(LTI) dynamics is represented by a state equation:

$$\dot{x}(t) = \mathbf{A}x(t) + \mathbf{B}u(t) \tag{1}$$

In this equation, vector $x(t) = (x_1(t), x_2(t), \ldots, x_N(t))^T$ captures the state of the system of N nodes at time t; $\mathbf{A}$ is the $N \times N$ matrix describing the wiring topology and the interaction among N system nodes; $\mathbf{B}$ is the $N \times M$ ($M \leq N$) input matrix containing the nodes driven by M external inputs, input vector $u(t) = (u_1(t), u_2(t), \ldots, u_M(t))^T$ holds the external inputs at time t to drive the system. Any system described by (1) is controllable if and only if the $N \times NM$ matrix $\mathbf{C} = [B, AB, A^2B, \ldots, A^{N-1}B]$ has full rank, noted by rank$(\mathbf{C}) = N$ [7].

However, for any system of (1) in reality, most entries of $\mathbf{A}$ and $\mathbf{B}$ are only known by approximation except for zero entries [8]. Besides, calculating the rank of any matrix $\mathbf{C}$ requires $2^N - 1$ combinations [5], which would be impossible for a large system. Both constrains prevent against using the rank condition to verify the controllability of a given LTI system. In order to avoid these two constrains and also figure out whether any system described by (1) is controllable or not, Lin [8] raised structural controllability :

Definition 1: **Structural Controllability** [8] A system described by (1) is structural controllable iff there exists a completely controllable system having the same structure as it.

This definition implies, an uncontrollable system of (1) will be controllable if we properly change the value of some entries of $\mathbf{A}$ or $\mathbf{B}$ of (1). Additionally, necessary and sufficicent conditions of structural conrollability are given via system's graph representation. Given a system described by (1), a digraph $G(\mathbf{A}, \mathbf{B}) = (V_1 \cup V_2, E_1 \cup E_2)$, where $V_1 \cap V_2 = \emptyset$, could be obtained by a bijection ω. Specifically, for any element $b_{ij} \neq 0$ and $b_{ij} \in \mathbf{B}$, $a_{ij} \neq 0$ and $a_{ij} \in \mathbf{A}$, $\omega : a_{ij} \rightarrow \overrightarrow{\langle x_j, x_i\rangle}, \overrightarrow{\langle x_j, x_i\rangle} \in E_A, x_i, x_j \in V_1$ and $\omega : b_{ij} \rightarrow \overrightarrow{\langle u_j, x_i\rangle}, \overrightarrow{\langle u_j, x_i\rangle} \in E_2, u_j \in V_2$, in which any $u_j \in V_2$ obtained above is an external input. Then, few necessary definitions are below:

Definition 2: **Inaccessibility** [8] In $G(\mathbf{A}, \mathbf{B})$, any $x_i \in V_1$ is inaccessible if there is no directed path from any vertex $u_j \in V_2$.

Definition 3: **Dilation of Digraphs** [8] Any $G(\mathbf{A}, \mathbf{B})$ includes two kinds of vertex sets, $S \subseteq V_1$, and $T(S) = \{x_j | \overrightarrow{\langle x_j, x_i\rangle} \in E_1 \cup E_2, x_i \in S\}$. When $G(\mathbf{A}, \mathbf{B})$ contains a dilation iff $|S| > |T(S)|$, in which $|S|$ and $|T(S)|$ represent the cardinality of S and $T(S)$.

Definition 4: **Stem and Bud** [8] For a digraph such as $G(\mathbf{A}, \mathbf{B})$, a stem is a directed path such as $\{\overrightarrow{\langle x_1, x_2\rangle}, \overrightarrow{\langle x_2, x_3\rangle}, \ldots, \overrightarrow{\langle x_j, x_i\rangle}\}$. A bud is a directed cycle such as $\{\overrightarrow{\langle x_1, x_2\rangle}, \overrightarrow{\langle x_2, x_3\rangle}, \ldots, \overrightarrow{\langle x_j, x_1\rangle}\}$ plus an arc sharing its head with this cycle and this arc is called distinguished edge.

Definition 5: **Cactus** [8] By definition 4, any stem is a cactus. Besides, a stem S_0 and buds $B_1, B_2, \ldots, B_l$, then, $S_0 \cup B_1 \cup B_2 \cup \ldots B_l$ is a cactus if for any $i (1 \leq i \leq l)$ the tail of the distinguished edge of B_i is not the top vertex of S_0 but is the only common vertex of $S_0 \cup B_1 \cup B_2 \cup \ldots B_{i-1}$. A set of vertex-disjoint cacti is called a cactus.

Now, the necessary and sufficicent conditions of structural controllability are given below:

Theorem 1: **Lin's Structural Controllability Theorem** [8] The following three statements are equivalent:

1) A system of (1) is structurally controllable.
2) a) $G(\mathbf{A}, \mathbf{B})$ contains no inaccessible nodes.
   b) $G(\mathbf{A}, \mathbf{B})$ contains no dilation.
3) $G(\mathbf{A}, \mathbf{B})$ is spanned by cacti.

Particularly, structural controllable systems can be expressed to be controllable for almost all values of entries of $\mathbf{A}$ and $\mathbf{B}$ of (1) except for some pathological cases with certain constrains [5]. For example, an LTI system contains nodes $\{x_1, x_2, x_3, x_4\}$ with one external input u_1, and arcs $\{\overrightarrow{\langle u_1, x_1 \rangle}, \overrightarrow{\langle x_1, x_2 \rangle}, \overrightarrow{\langle x_2, x_3 \rangle}, \overrightarrow{\langle x_3, x_2 \rangle}, \overrightarrow{\langle x_3, x_4 \rangle}\}$. According to theorem 1, this system is structurally controllable because its digraph contains neither inaccessible nodes nor dilation, but its digraph would be not completely controllable if any strength among its all vertices is one, causing the rank condition [38] [7] be dissatified. Nevertheless, if this system delets the directed interaction $\overrightarrow{\langle x_3, x_2 \rangle}$, its current digraph would be always completely controllable by u_1, because rank of matrix $\mathbf{C}$ is now independent with the value of its any non-zero entry.

Except for pathological cases, Liu *et al* [5] proved that the maximum matching determines the minimum external inputs required to maintain full control of a digraph in LTI systems according to that controllability rank condition. With a digraph, noted by $G(\mathbf{A}) = (V_1, E_1)$, where V_1 and E_1 have been defined above, is fully controllable if and only if all unmatched nodes are directly controlled by external inputs and all matched nodes can be visited by inputs along with directed paths [39]. In other words, it is generalized by the minimum input theorem:

Theorem 2: **Minimum Input Theorem** [5] The minimum number of inputs to fully control a digraph such as $G(\mathbf{A}) = (V_1, E_1)$ is one if there is a perfect matching in $G(\mathbf{A})$. Otherwise, it is equal to the number of unmatched nodes according to any maximum matching.

When each node of a graph is matched, it is said that this graph contains a perfect matching.

IV. Preliminaries

In this section, firstly, we define a kind of input digraphs, in which we assume that each input digraph is generated by adding a node with relative arcs to a same type of digraphs. We propose to efficiently identify a maximum matching without recomputation.

Definition 6: We consider any large, sparse *Erdős-Rényi* random digraph that excludes isolated vertices, parallel arcs and selfloops as our input graph, noted by $D = (V, E)$, where vertex set $V = \{v_i | 1 \leq i \leq N\} (N > 1)$ and arc set $E = \{\overrightarrow{\langle v_i, v_j \rangle} | 1 \leq i, j \leq N, i \neq j\}$. A known matching of D is noted by M_0.

Besides, any $D = (V, E)$ is obtained by adding a node noted by u with relative arcs to an original digraph, noted by $D_0 = (V_0, E_0)$. The probability of edges existence between u and D_0 is same as that of D_0. In particular, D_0 is also assumed to contain a known maximum matching noted by M_0. A maximum matching of D proposed to be identified is noted by M, and it is possible that $M = M_0$.

Then, we use a directed biaprtite representation of D to find M, which is defined below:

Definition 7: Given $D = (V, E)$ of definition 6, a bipartite graph $B = (V_B, E_B)$ $|E_B| = |E|$, $V_B = V_B^+ \cup V_B^-$, $V_B^+ \cap V_B^- = \emptyset$ is derived by two bijections: α, β. For every arc $\overrightarrow{\langle v_i, v_j \rangle} \in E - M_0$, $\overrightarrow{\langle v_i, v_j \rangle} \notin \emptyset$, $\alpha : \overrightarrow{\langle v_i, v_j \rangle} \rightarrow \overrightarrow{\langle v_i^+, v_j^- \rangle}$, $\overrightarrow{\langle v_i^+, v_j^- \rangle} \in E_B$, $v_i^+ \in V_B^+$, and $v_j^- \in V_B^-$. A matching mapped from M_0 of D is noted by M_{B_0}. For each arc $\overrightarrow{\langle v_x, v_y \rangle} \in M_0$, $\overrightarrow{\langle v_x, v_y \rangle} \notin \emptyset$, $\beta : \overrightarrow{\langle v_x, v_y \rangle} \rightarrow \overrightarrow{\langle v_y^-, v_x^+ \rangle}$, $\overrightarrow{\langle v_y^-, v_x^+ \rangle} \in M_{B_0}$, $v_x^+ \in V_B^+$, $v_y^- \in V_B^-$.

Therefore, identifying a maximum matching of D is transferred by finding a maximum matching in B. By definition 7 and the added node $u \in D$ to D_0, we could also obtain at most two nodes correspond to u. We note them as $u^+ \in V_B^+$ and $u^- \in V_B^-$. Whether $u^+ \in \emptyset$ and $u^- \in \emptyset$ or not depends on whether u is a head or a tail of its incident arcs.

Regarding to finding a maximum matching of a bipartite graph, it is indispensible to mention the augmenting path [18] [19] here. With $B = (V_B, E_B)$ and a matching $M_{B_0} \in B$, the augmenting path can be defined:

*Definition 8 (**Augmenting path** [19] [18]):* Within $B = (V_B, E_B)$ of definition 7, an alternating path with respect to M_{B_0} alternatively contains the edges in M_{B_0} and $E_B - M_{B_0}$. When the both top and bottom nodes of this alternating path are only out of M_{B_0}, it is called an augmenting path relative to M_{B_0}.

Here are few very important proved propositions about the augmenting path and matching:

Proposition 1: By definition 8, if $M_{B_0} \in E_B$ is a matching and P_a is an augmenting path relative to it, then the symmetric difference of M_{B_0} and P_a represented by $M_{B_0} \oplus P_a$, is a bigger matching than M_{B_0} in cardinality by one [19] [18].

Proposition 2: By definition 8, $M_{B_0} \in E_B$ is a maximum matching of B iff there is no augmenting path relative to M_{B_0} [19] [18].

V. Recover controllability of D

A. Identify a maximm matching of B

It is assumed that $D = (V, E)$ of definition 6 is obtained by adding u to D_0, we can also assume that $B = (V_B, E_B)$ is obtained by adding u^+ or u^- to a bipartite graph, noted by $B_0 = (V_{B_0}, E_{B_0})$. Then, a maximum matching of B_0 is noted by M_{B_0} in following paper.

Since M_{B_0} is the maximum matching of B_0, and adding u^- or u^+ with relative edges generates B. Obviously, augmenting paths relative to M_{B_0} can be only incident to u^- or u^+. Nevertheless, when both u^+ and u^- exisit in V_B^- and V_B^+, there can not be an edge connecting u^+ and u^-, otherwise there would be selfloop in D, which is contradicted with D without selfloop of definition 6.

The following algorithm finds augmenting paths starting from u^+ and ending at a node of V_B^- out of M_{B_0} in $B = (V_B, E_B)$ when $u^+ \notin \emptyset$. An edge set noted by T_{u^+} whose tails are u^+. Any edge of T_{u^+} is noted by $e \in T_{u^+}$, P_0 represents an edge set, and $P(P_0)$ dentoes a set including edges of E_B pointed by edges of P_0. For any edge of $P(P_0)$ is noted by $e' \in P(P_0)$ and $e' = \overrightarrow{\langle v_x^+, v_y^- \rangle}$. Besides, P_{sub} denotes any subpath of a returned augmenting path ending at a node of V_B^- and out of M_{B_0}, and P_n notes a path starting from u^+ and ending at one node of M_{B_0}. Besides, we use P_a^+ to denote any augmenting path incident to u^+.

Algorithm 1 Find Augmenting Paths incident to u^+

Input: $B = (V_B, E_B)$, M_{B_0}, $u^+ \notin \emptyset$
Output: Augmenting Paths incident to u^+
1: $P_0 = \emptyset$
2: **while** $T_{u^+} \neq \emptyset$ **and** $e \in T_{u^+}$ **do**
3: $T_{u^+} = T_{u^+} - e$
4: **if** two terminals of e out of M_{B_0} **or** e pointing P_{sub} **then**
5: **return** $P_a^+ = e$ **or** $P_a^+ = \{e, P_{sub}\}$
6: **else if** e just pointing a node of M_{B_0} **then**
7: $P_0 = P_0 + e$
8: **for** $P(P_0) \in E_B$ **and** $e' \in P(P_0)$ **do**
9: $P_0 = P_0 + e'$ **and** $E_B = E_B - e'$
10: **if** e' pointing P_n **and** v_y^- out of M_{B_0} **then**
11: **return** $P_a^+ = \{P_n, e'\}$
12: **else if** e' pointed by P_n **and** pointing P_{sub} **then**
13: **return** $P_a^+ = \{P_n, e', P_{sub}\}$

Proof: Initially, since $u^+ \notin \emptyset$, T_{u^+} can not be empty. If e is not adjacent to any edge of M_{B_0}, e is an augmenting path by definition 8 and returned. If not, e of T_{u^+} is added into P_0 to find augmenting paths involving from line 8-13. In detail, any e' of $P(P_0)$ pointed by e is concerned in line 10 and then added into P_0 and removed from E_B to guarantee that each edge of E_B is considered at most once. Since by now $e' \in M_{B_0}$, e and e' can not be an augmenting path. Thus, keep visiting edges pointed by current P_0 until an edge of E_B whose head is out of M_{B_0}, and such a path from u^+ is an augmenting path involving P_n and e'. Because several augmenting paths from u^+ may be vertex joint or overlapped, we just need to check whether an edge is poniting a subpath of a known augmenting path or not rather than revisiting some edges. Once $P(P_0) = \emptyset$, it means that search of all paths starting from e is complete. After that, the following edge of T_{u^+} is one by one considered. For the same reason, following augmenting path starting from this newly added edge of T_{u^+} may be overlapped with reviously returned ones. Therefore, new edge of T_{u^+} needs to be checked whether it is pointing a P_{sub} of an augmenting path in line 4. If so, it is directly returned. Otherwise, it would construct a new augmenting path with P_n. Finally, since each edge of T_{u^+} is removed in line 3, we would obtain $T_{u^+} = \emptyset$, when this procedure terminates. For time complexity, it depends on the cardinality of T_{u^+} and the number of edges of P_0. Thus, it can be represented by $O(|E_B|)$ or $O(|E|)$ by definition 7. ∎

When $u^- \notin \emptyset$, this algorithm can also find augmenting paths starting from a node out of M_{B_0} of V_B^+ and ending at u^- in $O(|E|)$, where T_{u^+} should be replaced with H_{u^-} meaning a set of edges whose head is u^-. $P(P_0)$ would be a set of edges pointing edges of P_0, P_{sub} now starting from a node of V_B^+ out of M_{B_0} and pointing e or e', and P_n starting from a node of M_{B_0} and ending at u^-. P_a^- represnets any augmenting path incident to u^-.

If two augmenting paths incident to u^+ and u^- respectively are vertex joint, we can not use them to derive any maximum matching.

Accordingly, the following algorithm identifies vertex-joint augmenting paths when $u^+ \notin \emptyset$ and $u^- \notin \emptyset$. In this algorithm, S_a is a subgraph of B, having all found augmenting paths, and S_b denotes nodes of S_a, noted by $S_b = N(S_a)$. For any $v^* \in S_b$, we use $\delta(v^*)$ to represent the number of adjacent nodes of S_b to v^*. We also use S_c to represent the set of vertex-joint augmenting paths, and C_{S_a} denotes a component of S_a.

Algorithm 2 Find vertex-joint augmenting paths

Input: S_a, S_b
Output: vertex-joint P_a^+ and P_a^-
1: $S_c = \emptyset$ **and** $S_b = S_b - u^- - u^+$
2: **for** $S_b \neq \emptyset$ **and** $v^* \in S_b$ **do**
3: $S_b = S_b - v^*$
4: **if** $\delta(v^*) > 2$ **then**
5: Find vertex-joint P_a^+ and P_a^- in C_{S_a} containing v^*
6: $S_c = S_c + P_a^+ + P_a^-$ **and** $S_a = S_a - C_{S_a}$
7: $S_b = N(S_a)$
8: **return** S_c

Proof: Except u^- and u^+, $\delta(v^*) > 2$ means v^* of S_b is shared by multiple augmenting paths, we then find vertex-joint P_a^- and P_a^+. Traversing arcs only once of C_{S_a} of S_a that contains v^* until there is no arc can operate procedure of line 5. It means that each traversed arc is connected to v^* based on the underlying undirected graph of S_a. Then, those visited paths during traverse of C_{S_a} must be vertex-joint, and we can immediately known which P_a^+ is vertex joint to which P_a^-. Identified vertex-joint P_a^- and P_a^+ would be added into S_c, while C_{S_a} is removed from S_a to guarantee that each arc of S_a is traversed once at most in line 5. After this, remaining nodes of S_a would be checked and further find vertex-joint augmenting paths by finding shared nodes in the first place. Because each checked node of S_b is removed, and cardinality of S_b is reduced along with reduce of S_a in line 6,7, this algorithm would terminate when $S_b = \emptyset$. For time complexity, it depends on $|N_A|$ and S_A indicated by line 2, 5. Since $|E_B| = |E|$, $|V_B| \leq 2|V|$ by definition 7, time complexity of this algorithm is represented by $O(|V|+|E|)$. ■

After obtaining available augmenting paths incident to u^- or u^+, and also obtaining all vertex-joint augmenting paths. The next algorithm determines a maximum matching of B under all possible cases.

Algorithm 3 Obtain a Maximum Matching of B

Input: M_{B_0}, S_a, S_c
Output: A maximum matching of B
1: **if** $\exists\ P_a^-, P_a^+$ **and** P_a^-, P_a^+ vertex disjoint **then**
2:     **return** $M_{B_0} \bigoplus P_a^- \bigoplus P_a^+$
3: **else if** $\exists\ P_a^-, P_a^+$ **and** P_a^-, P_a^+ vertex joint **then**
4:     **return** $M_{B_0} \bigoplus P_a^-$ **or** $M_{B_0} \bigoplus P_a^+$
5: **else if** $S_c = \emptyset$ **and** $\exists\ P_a^+ = P_a^-$ **then**
6:     **return** $M_{B_0} \bigoplus P_a^-$ **or** $M_{B_0} \bigoplus P_a^+$
7: **else if** $\nexists\ P_a^+$ **and** $\exists\ P_a^-$ **then**
8:     **return** $M_{B_0} \bigoplus P_a^-$
9: **else if** $\exists\ P_a^+$ **and** $\nexists\ P_a^-$ **then**
10:     **return** $M_{B_0} \bigoplus P_a^+$
11: **else if** $S_a = \emptyset$ **then**
12:     **return** M_{B_0}

Proof: According to proposition 1, 2, we know how to use the augmenting paths construct a bigger cardinality matching than a known one. Thus, with found augmenting paths by using **Algorithm 1** we could directly use the symmetric difference among M_{B_0}, any P_a^- or P_a^+ to obtain a maximum matching by augmenting M_{B_0}. Because any augmenting path of B is leaded by u^- and u^+, which can be seen as the effect of adding u^- and u^+ to B_0 as mentioned before. As a result, we classify all possible cases based on whether P_a^- or P_a^+ exists in B. Thus, it is possible that both P_a^- and P_a^+ exist and they might be vertex-joint augmenting paths or not. Such cases are given from line 1 to 5. Besides, there might be $P_a^- = \emptyset$ or $P_a^+ = \emptyset$, meaning augmenting paths starting from u^+ or ending at u^- do not exist, althouth u^- or u^+ exists in B. Based on cases related to this, lines of 7-12 give the other all possible results about a maximum matching of B. Since any P_a^+, P^- and vertex-joint P_a^+ and P^- have been known before by using **Algorithm 1, 2**. Therefore, there is no need of extra computation to confirm whether P_a^+ and P_a^- exist or not and whether they are vertex joint or not, time complexity of this procedure is therefore $O(1)$. ■

In following part, we denote M_B as the returned maximum matching of B by **Algorithm 3**. And we would identify a maximum matching of D of definition 6 in the next part.

B. Identify a maximum matching of D

Since our goal is to efficiently recover controllability of an input digraph $D = (V, E)$ of definition 1 via finding its maximum matching, rather than reusing the best-known algorithms of [15] and [16]. With the identified M_B of $B = (V_B, E_B)$ of definition 7, we now can directly obtain a maximum matching M of D by inverse of bijection α and β of definition 7 in following algorithm. Each edge of M_B is noted by $e_{M_B} \in M_B$, and it is either $e_{M_B} = \overrightarrow{\langle v_i^-, v_j^+ \rangle}$ or $e_{M_B} = \overrightarrow{\langle v_i^+, v_j^- \rangle}$.

Algorithm 4 Identify a maximum matching M of D

Input: $D = (V, E)$, M_{B_0}, M_B
Output: Maximum matching of D
1: $M = \emptyset$
2: **for** $M_B \neq \emptyset$ **and** each $e_{M_B} \in M_B$ **do**
3:     $M_B = M_B - e_{M_B}$
4:     **if** $e_{M_B} \notin M_{B_0}$ **and** $e_{M_B} = \overrightarrow{\langle v_i^+, v_j^- \rangle}$ **then**
5:         $\alpha^- : e_{M_B} \to \overrightarrow{\langle v_i, v_j \rangle}$ **and** $M = M + \overrightarrow{\langle v_i, v_j \rangle}$
6:     **else if** $e_{M_B} \in M_{B_0}$ **and** $e_{M_B} = \overrightarrow{\langle v_i^-, v_j^+ \rangle}$ **then**
7:         $\beta^- : e_{M_B} \to \overrightarrow{\langle v_j, v_i \rangle}$ **and** $M = M + \overrightarrow{\langle v_j, v_i \rangle}$
8: **return** M

Proof: Since each mapped edge of M_B is removed in line 3, M_B would be empty, and this procedure terminates. For time complexity, it is $\Theta(|M_B|)$ or $O(|E_B|)$. Due to $|E| = |E_B|$ by definition 7, time complexity of this algorithm is also $O(|E|)$. ■

C. Complexity Analysis

The whole process of our scenario can be represented:

Algorithm 5 Identify a maximum matching of D

Input: $D = (V, E)$, M_0
Output: A maximum matching of D
1: Generate the bipartite graph B by D.
2: Find all augmenting paths incident to u^- and u^+.
3: Distinguish vertex-joint augmenting paths incident to u^- and u^+.
4: Obtain M_B of B.
5: Identify M of D through M_B.
6: **return** M.

In this scenario, each line presents a procedure, in line 1, the procedure of obtaining the bipartite graph B by the digraph D can be finished in $\Theta(|E|)$ by definition 7. Then, from procedures of line 2 to line 5, they can be finished by previous linear **Algorithm 1-4** respectively. Therefore, the worst-case execution time of the whole process of obtaining a maximum matching of digraph $D = (V, E)$ of definition 6 is $O(|V| + |E|)$, concluded by plus time complexity of each procedure. As a result, we efficiently obtain a maximum matching of an input digraph D of definition 6, rather than recomputation of a maximum matching of D, and finally we acquire its controllability via the found maximum matching according to theorem 2.

Consequently, iterative recovery of controllability via maximum matching could be executed in an effective manner in practice, such as calculating control robustness against nodal remvoal attack.

VI. Conclusion

The structure of dymanical systems might be changed for different reasons, such as being attacked, extented by giving new vertices or random failure happens. As a result, embedded digraphs would be also changed by insertion or deletion of nodes or edges. In this situation, recomputation of a maximum matching is no longer effective to acquire controllability of each changed digraph for several times. Facing with this problem, we come up with a method of efficiently obtaining a maximum matching of each incremental digraph in linear time, so as to reduce the complexity of iterative computation of maximum matchings in the whole process. Our mehod mainly based on a bipartition representation of a given incremental digraph and find augmenting paths that are only incident to the nodes corresponding to the added nodes. Besides, we also distinguish whether found augmenting paths are vertex joint or not, so that we could eventually obtain a maximum matching of an incremental digraph and further obtain its controllability. For future work, concerning malicious attack changes controllability dramatically, it is essential and interesting to iteratively recover controllability after each attack such as a nodal removal.

References

[1] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of modern physics*, vol. 74, no. 1, p. 47, 2002.

[2] M. E. Newman, "The structure and function of complex networks," *SIAM review*, vol. 45, no. 2, pp. 167–256, 2003.

[3] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[4] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-worldnetworks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[5] Y.-Y. Liu, J.-J. Slotine, and A.-L. Barabási, "Controllability of complex networks," *Nature*, vol. 473, no. 7346, pp. 167–173, 2011.

[6] M. Doostmohammadian and U. A. Khan, "On the genericity properties in distributed estimation: Topology design and sensor placement," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 2, pp. 195–204, 2013.

[7] R. Kalman, "On the general theory of control systems," *Automatic Control, IRE Transactions on*, vol. 4, no. 3, pp. 110–110, 1959.

[8] C. T. Lin, "Structural controllability," *Automatic Control, IEEE Transactions on*, vol. 19, no. 3, pp. 201–208, 1974.

[9] G. Chartrand, L. Lesniak, and P. Zhang, *Graphs & digraphs*. CRC Press, 2010.

[10] B. Wang, L. Gao, Y. Gao, and Y. Deng, "Maintain the structural controllability under malicious attacks on directed networks," *EPL (Europhysics Letters)*, vol. 101, no. 5, p. 58003, 2013.

[11] J. Ruths and D. Ruths, "Robustness of network controllability under edge removal," in *Complex Networks IV*. Springer, 2013, pp. 185–193.

[12] S. Nie, X. Wang, H. Zhang, Q. Li, and B. Wang, "Robustness of controllability for networks based on edge-attack," *PloS one*, vol. 9, no. 2, p. e89066, 2014.

[13] H. Lvlin, L. Songyang, B. Jiang, and B. Liang, "Enhancing complex network controllability by rewiring links," in *Intelligent System Design and Engineering Applications (ISDEA), 2013 Third International Conference on*. IEEE, 2013, pp. 709–711.

[14] W.-X. Wang, X. Ni, Y.-C. Lai, and C. Grebogi, "Optimizing controllability of complex networks by minimum structural perturbations," *Physical Review E*, vol. 85, no. 2, p. 026115, 2012.

[15] J. E. Hopcroft and R. M. Karp, "An n^5/2 algorithm for maximum matchings in bipartite graphs," *SIAM Journal on computing*, vol. 2, no. 4, pp. 225–231, 1973.

[16] S. Micali and V. V. Vazirani, "An o (v— v— c— e—) algoithm for finding maximum matching in general graphs," in *Foundations of Computer Science, 1980., 21st Annual Symposium on*. IEEE, 1980, pp. 17–27.

[17] P. ERDdS and A. R&WI, "On random graphs i," *Publ. Math. Debrecen*, vol. 6, pp. 290–297, 1959.

[18] D. Luenberger, "Introduction to dynamic systems: theory, models, and applications," 1979.

[19] R. Z. Norman and M. O. Rabin, "An algorithm for a minimum cover of a graph," *Proceedings of the American Mathematical Society*, vol. 10, no. 2, pp. 315–319, 1959.

[20] T. W. Haynes, S. M. Hedetniemi, S. T. Hedetniemi, and M. A. Henning, "Domination in graphs applied to electric power networks," *SIAM Journal on Discrete Mathematics*, vol. 15, no. 4, pp. 519–529, 2002.

[21] J. Flum, "Downey rg and fellows mr. parameterized complexity. monographs in computer science. springer, new york, berlin, and heidelberg, 1999, xv+ 533 pp." *Bulletin of Symbolic Logic*, vol. 8, no. 04, pp. 528–529, 2002.

[22] A. Aazami and K. Stilp, "Approximation algorithms and hardness for domination with propagation," *SIAM Journal on Discrete Mathematics*, vol. 23, no. 3, pp. 1382–1399, 2009.

[23] J. Guo, R. Niedermeier, and D. Raible, "Improved algorithms and complexity results for power domination in graphs," *Algorithmica*, vol. 52, no. 2, pp. 177–202, 2008.

[24] B. Alwasel and S. D. Wolthusen, "Recovering structural controllability on erdős-rényi graphs via partial control structure re-use," in *International Conference on Critical Information Infrastructures Security*. Springer, 2014, pp. 293–307.

[25] ——, "Recovering structural controllability on erdős-rényi graphs in the presence of compromised nodes," in *International Conference on Critical Information Infrastructures Security*. Springer, 2015, pp. 105–119.

[26] ——, "Structural controllability analysis via embedding power dominating set approximation in erdhos-rènyi graphs," in *Advanced Information Networking and Applications Workshops (WAINA), 2015 IEEE 29th International Conference on*. IEEE, 2015, pp. 418–423.

[27] C. Alcaraz and S. Wolthusen, "Recovery of structural controllability for control systems," in *International Conference on Critical Infrastructure Protection*. Springer, 2014, pp. 47–63.

[28] R. Duan and S. Pettie, "Linear-time approximation for maximum weight matching," *Journal of the ACM (JACM)*, vol. 61, no. 1, p. 1, 2014.

[29] H. Alt, N. Blum, K. Mehlhorn, and M. Paul, "Computing a maximum cardinality matching in a bipartite graph in time o (n 1.5 mlog n)," *Information Processing Letters*, vol. 37, no. 4, pp. 237–240, 1991.

[30] M. Gupta and R. Peng, "Fully dynamic (1+ e)-approximate matchings," in *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*. IEEE, 2013, pp. 548–557.

[31] P. Zhang, *Handbook of graph theory*. Chapman and Hall/CRC, 2013.

[32] K. Onak and R. Rubinfeld, "Maintaining a large matching and a small vertex cover," in *Proceedings of the forty-second ACM symposium on Theory of computing*. ACM, 2010, pp. 457–464.

[33] S. Baswana, M. Gupta, and S. Sen, "Fully dynamic maximal matching in o(\logn) update time," *SIAM Journal on Computing*, vol. 44, no. 1, pp. 88–113, 2015.

[34] A. Bernstein and C. Stein, "Faster fully dynamic matchings with small approximation ratios," in *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2016, pp. 692–711.

[35] S. Bhattacharya, M. Henzinger, and D. Nanongkai, "Fully dynamic approximate maximum matching and minimum vertex cover in o (log3 n) worst case update time," in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2017, pp. 470–489.

[36] P. Sankowski, "Faster dynamic matchings and vertex connectivity," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 118–126.

[37] J.-J. E. Slotine, W. Li, *et al.*, *Applied nonlinear control*. prentice-Hall Englewood Cliffs, NJ, 1991, vol. 199, no. 1.

[38] R. E. Kalman, "Mathematical description of linear dynamical systems," *Journal of the Society for Industrial and Applied Mathematics, Series A: Control*, vol. 1, no. 2, pp. 152–192, 1963.

[39] W. Yu, G. Chen, M. Cao, and J. Kurths, "Second-order consensus for multiagent systems with directed topologies and nonlinear dynamics," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 40, no. 3, pp. 881–891, 2010.